

CS6910 Assignment - 1

Name : Chathurvedhi Talapaneni

Roll No : CS20B021

Wandb Report : [link](#)

Github repo : [link](#)

CS6910 _CS20B021- Assignment_1

Write your own backpropagation code and keep track of your experiments using wandb.ai

Chathurvedhit

▼ Instructions

- The goal of this assignment is twofold: (i) implement and use gradient descent (and its variants) with backpropagation for a classification task (ii) get familiar with Wandb which is a cool tool for running and keeping track of a large number of experiments
- This is a **individual assignment** and no groups are allowed.
- Collaborations and discussions with other students is strictly prohibited.
- You must use Python (NumPy and Pandas) for your implementation.
- You cannot use the following packages from Keras, PyTorch, Tensorflow: optimizers, layers
- If you are using any packages from Keras, PyTorch, Tensorflow then post on Moodle first to check with the instructor.
- You have to generate the report in the same format as shown below using wandb.ai. You can start by cloning this report using the clone option above. Most of the plots that we have asked for below can be (automatically) generated using the APIs provided by `wandb.ai`. You will upload a link to this report on Gradescope.

- You also need to provide a link to your GitHub code as shown below. Follow good software engineering practices and set up a GitHub repo for the project on Day 1. Please do not write all code on your local machine and push everything to GitHub on the last day. The commits in GitHub should reflect how the code has evolved during the course of the assignment.
- You have to check Moodle regularly for updates regarding the assignment.

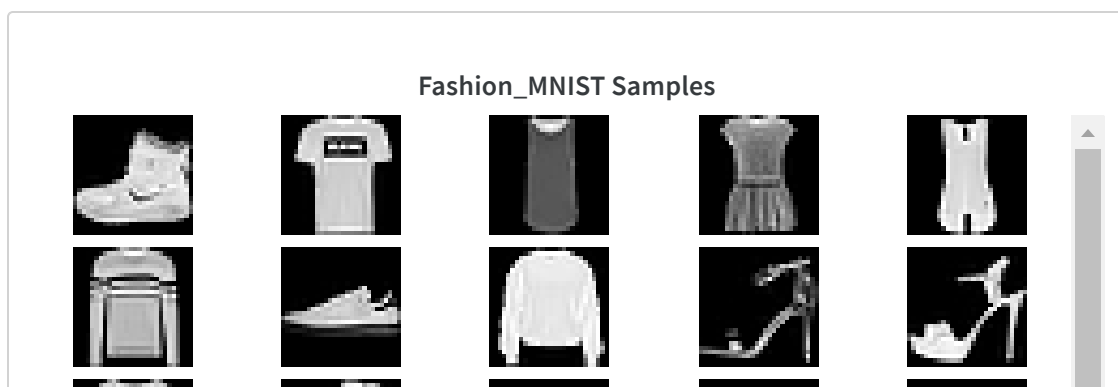
▼ Problem Statement

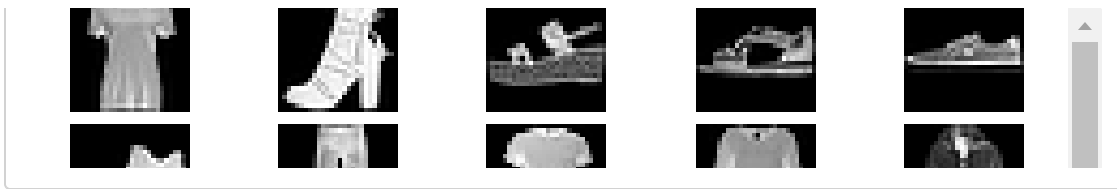
In this assignment you need to implement a feedforward neural network and write the backpropagation code for training the network. We strongly recommend using numpy for all matrix/vector operations. You are not allowed to use any automatic differentiation packages. This network will be trained and tested using the Fashion-MNIST dataset. Specifically, given an input image (28 x 28 = 784 pixels) from the Fashion-MNIST dataset, the network will be trained to classify the image into 1 of 10 classes.

Your code will have to follow the format specified in the `Code Specifications` section.

▼ Question 1 (2 Marks)

Download the fashion-MNIST dataset and plot 1 sample image for each class as shown in the grid below. Use `from keras.datasets import fashion_mnist` for getting the fashion mnist dataset.





▼ Question 2 (10 Marks)

Implement a feedforward neural network which takes images from the fashion-mnist data as input and outputs a probability distribution over the 10 classes.

Your code should be flexible such that it is easy to change the number of hidden layers and the number of neurons in each hidden layer.

▼ Question 3 (24 Marks)

Implement the backpropagation algorithm with support for the following optimisation functions

- sgd
- momentum based gradient descent
- nesterov accelerated gradient descent
- rmsprop
- adam
- nadam

(12 marks for the backpropagation framework and 2 marks for each of the optimisation algorithms above)

We will check the code for implementation and ease of use (e.g., how easy it is to add a new optimisation algorithm such as Eve). Note that the code should be flexible enough to work with different batch sizes.

Question 4 (10 Marks)

Use the sweep functionality provided by wandb to find the best values for the hyperparameters listed below. Use the standard train/test split of fashion_mnist (use `(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()`). Keep 10% of the training data aside as validation data for this hyperparameter search. Here are some suggestions for different values to try for hyperparameters. As you can quickly see that this leads to an exponential number of combinations. You will have to think about strategies to do this hyperparameter search efficiently. Check out the options provided by `wandb.sweep` and write down what strategy you chose and why.

- number of epochs: 5, 10
- number of hidden layers: 3, 4, 5
- size of every hidden layer: 32, 64, 128
- weight decay (L2 regularisation): 0, 0.0005, 0.5
- learning rate: 1e-3, 1 e-4
- optimizer: sgd, momentum, nesterov, rmsprop, adam, nadam
- batch size: 16, 32, 64
- weight initialisation: random, Xavier
- activation functions: sigmoid, tanh, ReLU

wandb will automatically generate the following plots. Paste these plots below using the "Add Panel to Report" feature. Make sure you use meaningful names for each sweep (e.g. `hl_3_bs_16_ac_tanh` to indicate that there were 3 hidden layers, batch size was 16 and activation function was ReLU) instead of using the default names (whole-sweep, kind-sweep) given by wandb.

Names of sweep were edited to meaningful names for the top 50 runs of all sweeps.

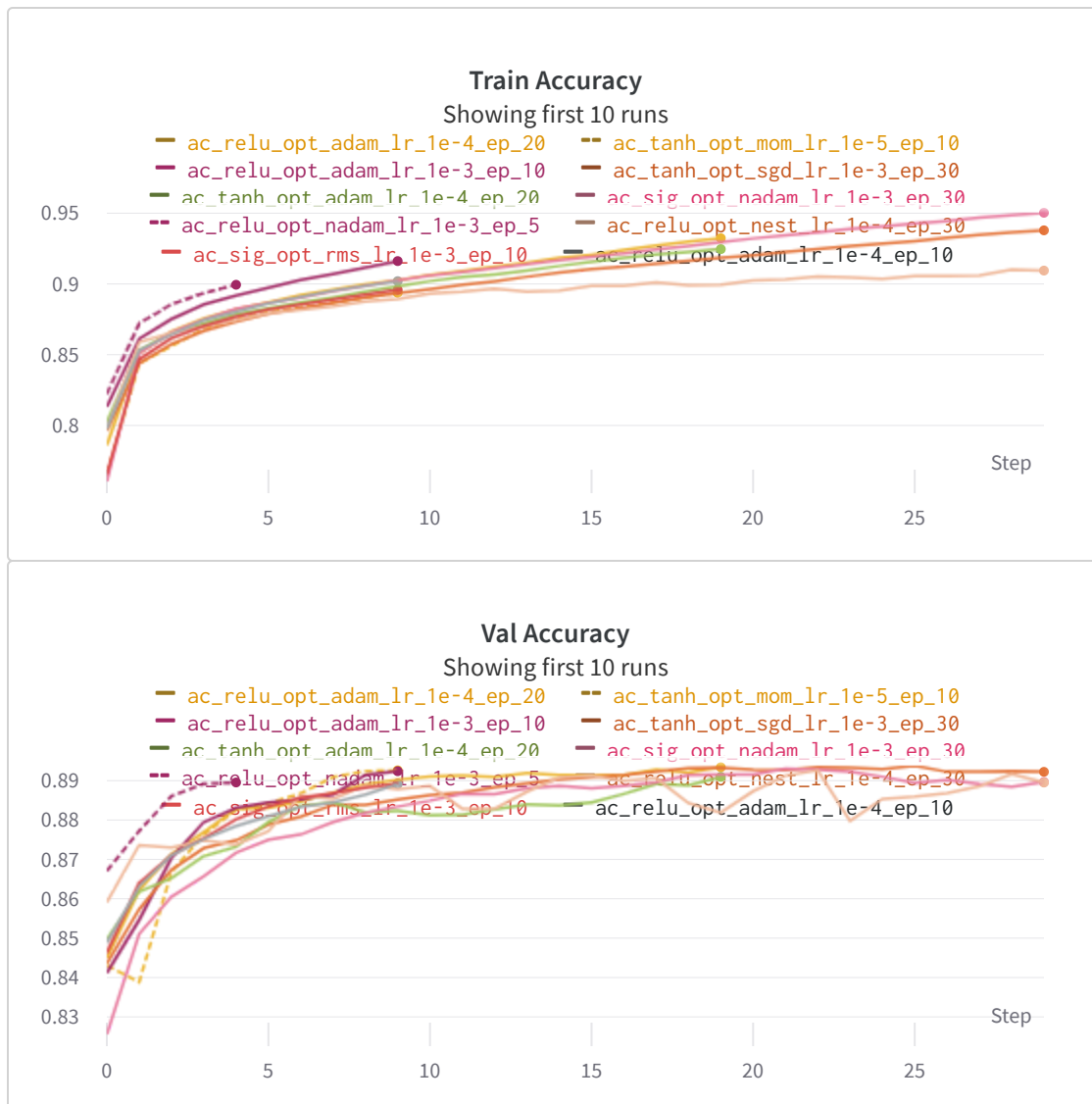
The choosing of Hyper parameters was done in 3 sweeps

Sweep 1

All possible optimizers and activations were chosen with varying other hyperparameters as well.

Outcomes:

- Weight Initialization chosen : Xavier
- Identity Activation removed due to very low performance
- SGD, Momentum, Nesterov Optimizers removed due to low performance
- Negative Correlation for Beta1 and learning rate

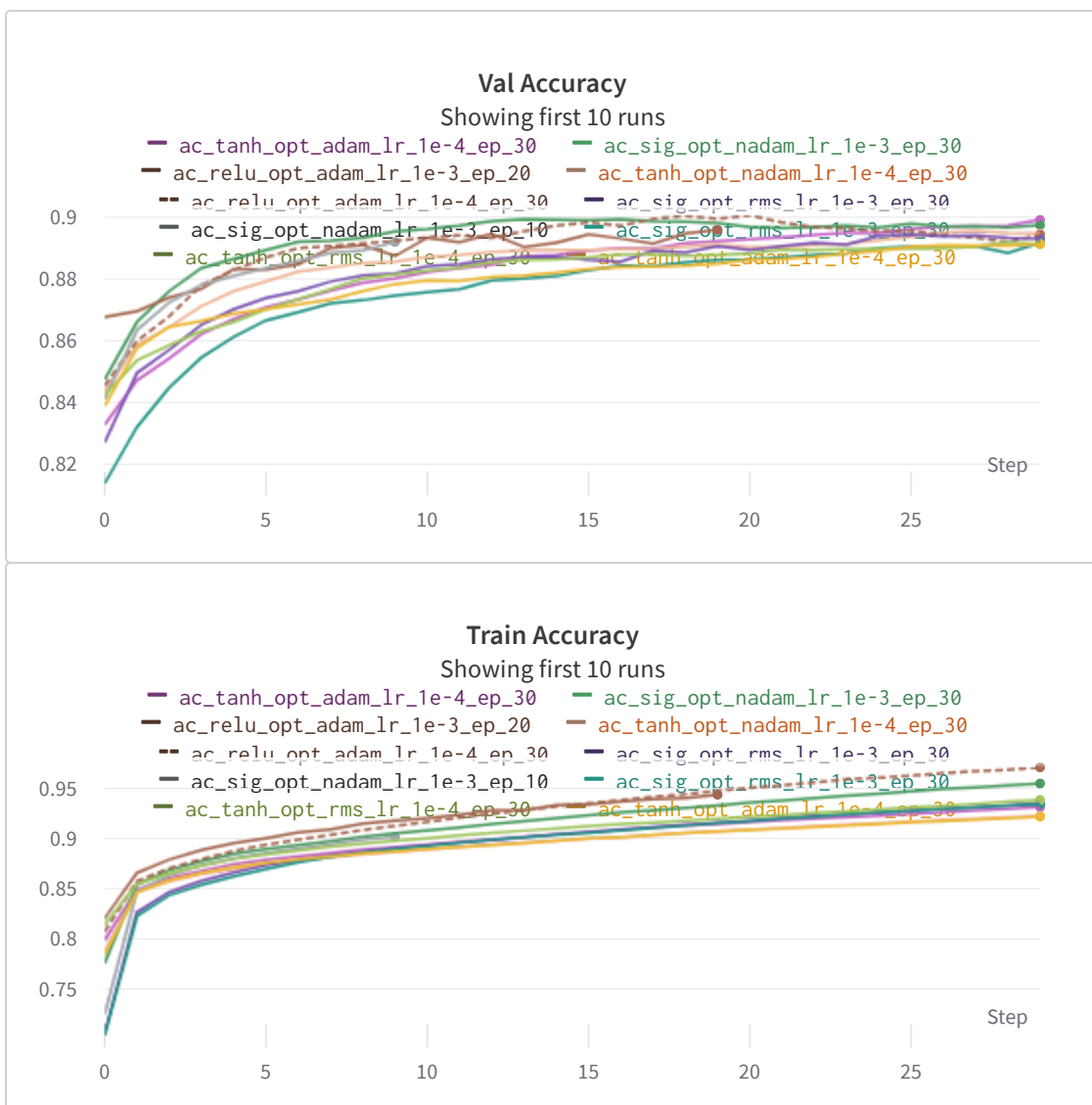


Sweep 2

Gave more possible values for epochs, batch size, hidden layers.

Outcomes:

- Sigmoid Activation Removed
- RMSprop Activation Removed
- Beta1 = 0.9, Beta2 = 0.999
- Lesser number of hidden layers and higher number of batch size gives better accuracy from Correlation plot.
- Weight decay, Layer size, epsilon have little to no significance

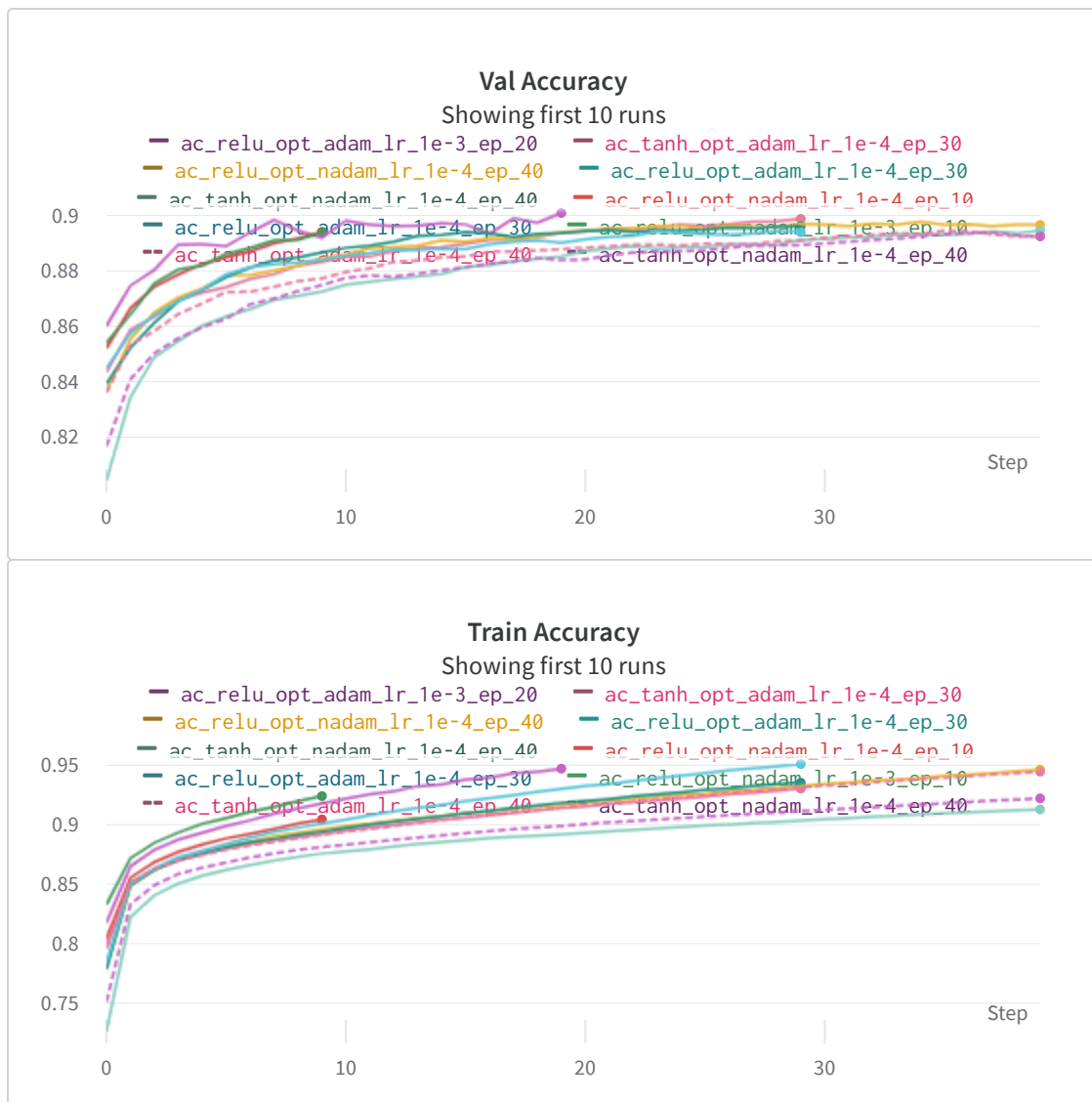


Sweep 3

Final Sweep for other parameters

Outcomes:

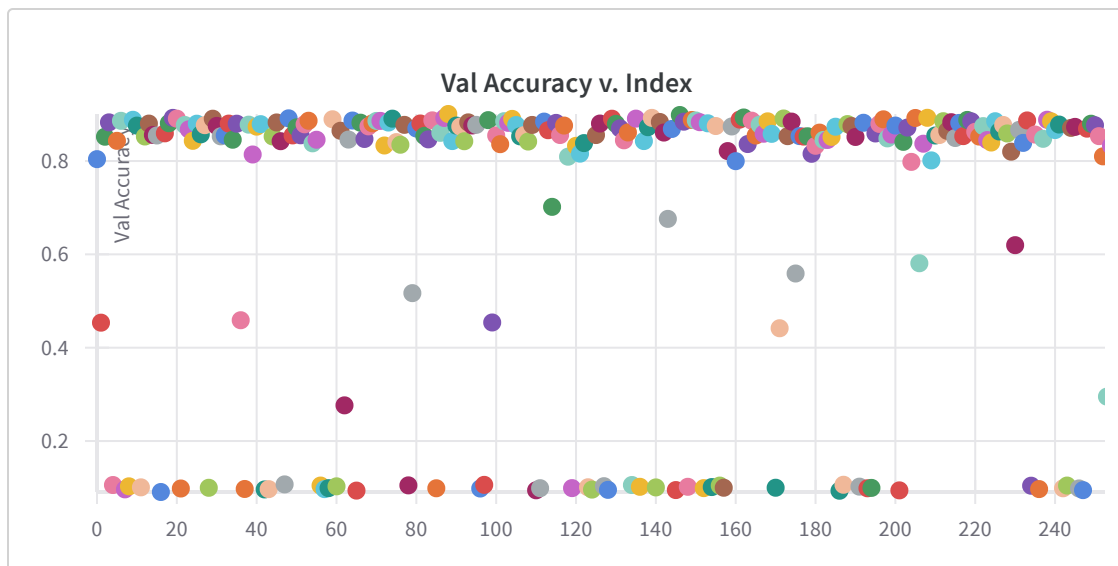
- Learning Rates : 0.001, 0.0001
- Epochs : 20, 30, 40
- Batch Size : 128, 256
- Number of Layers : 3, 4 (Hidden + 1)



▼ Question 5 (5 marks)

We would like to see the best accuracy on the validation set across all the models that you train.

wandb automatically generates this plot which summarises the test accuracy of all the models that you tested. Please paste this plot below using the "Add Panel to Report" feature



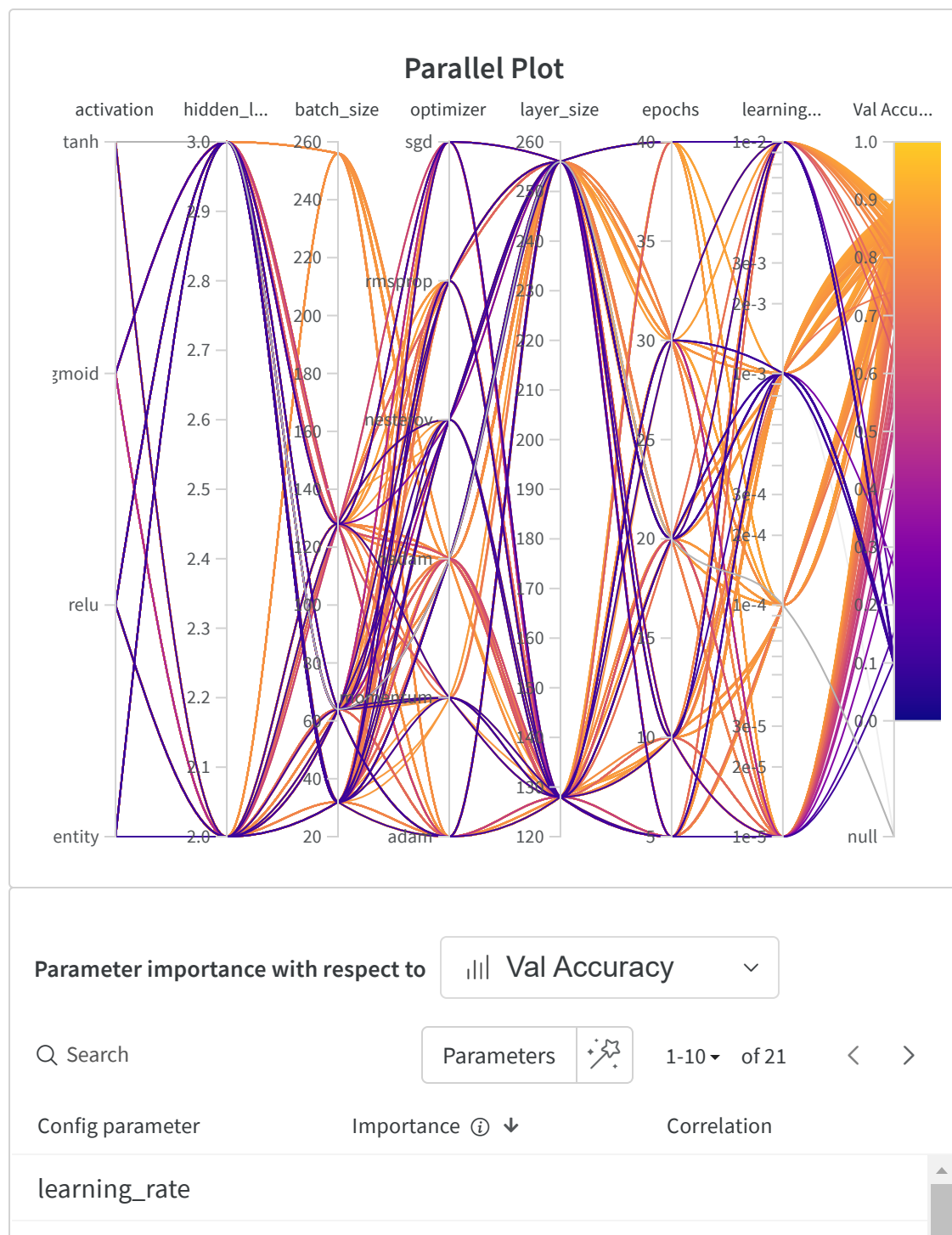
▼ Question 6 (20 Marks)

Based on the different experiments that you have run we want you to make some inferences about which configurations worked and which did not.

Here again, wandb automatically generates a "Parallel co-ordinates plot" and a "correlation summary" as shown below. Learn about a "Parallel co-ordinates plot" and how to read it.

By looking at the plots that you get, write down some interesting observations (simple bullet points but should be insightful). You can also refer to the plot in Question 5 while writing these insights. For example, in the above sample plot there are many configurations which give less than 65% accuracy. I would like to zoom into those and see what is happening.

I would also like to see a recommendation for what configuration to use to get close to 95% accuracy.



beta1
optimizer.value_nest...
optimizer.value_mo...
batch_size
hidden_layers

Inferences :

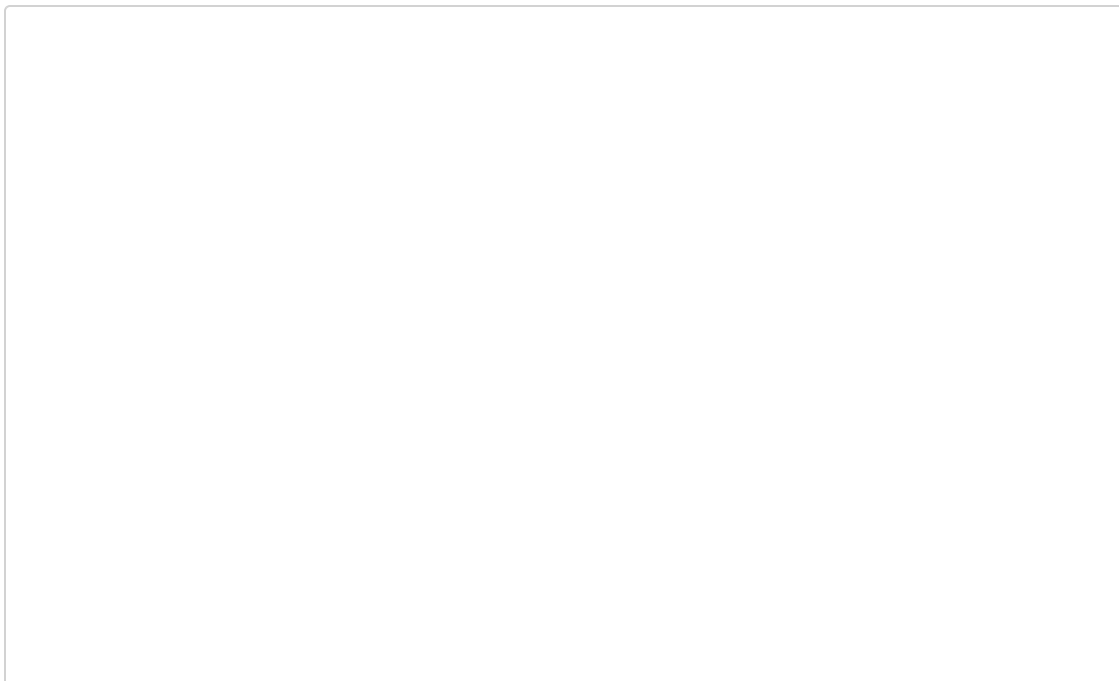
- The very low validation accuracies (or crashed) occur for following reasons
 1. If the optimizer is 'nesterov' or 'momnetum' with high learning rates. This leads to crashed runs recommending low learning rates for them.
 2. If the optimizer is 'sgd' with low learning rates. Thus we recommend higher learning rates.
- For the runs with Val Accuracy between 40% and 70%:
 1. The learning rates for most of these runs is often very low.
 2. Less number of Epochs(majority having 5 in the range) also leads to low Accuracy values.
- For the segment with very high accuracy
 1. Activation Func : Mostly dominated by Relu and Tanh, few Sigmoid runs but nearly zero Identity.
 2. Optimizer : Mostly dominated by Adam and Nadam, with a few RMSprop but the rest are very low in comparision.
 3. Epochs : Mostly 20, 30 number of epochs and few 40. Shown with more ep_10 values but there were more tests with ep_10 configuration.
- Also noticed that weight initialization as xavier provides much better results.

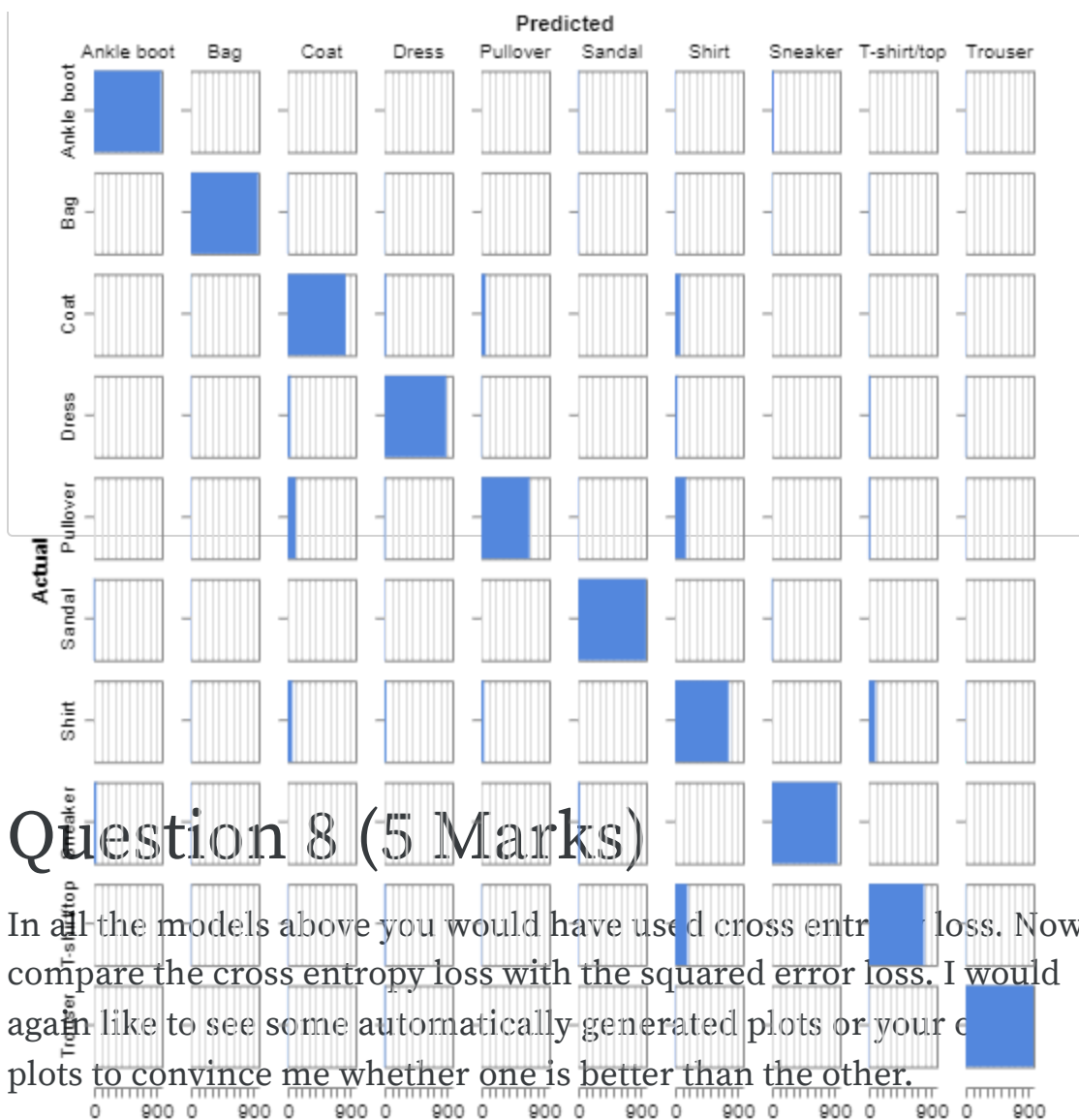
- To achieve near 95% accuracy:
 1. We must use Adam/Nadam along with Relu/Tanh as they show the most promise.
 2. On observing correlation for the very high accuracy values we notice that there is a positive correspondence to number of epochs.
 3. Many runs with high epoch values have very high Train Accuracies but Validation Accuracy doesn't reach that high.
 4. Thus we may be able to reach higher accuracies if we have better regularization along with higher number of epochs so that we avoid overfitting.

▼ Question 7 (10 Marks)

For the best model identified above, report the accuracy on the test set of fashion_mnist and plot the confusion matrix as shown below. More marks for creativity (less marks for producing the plot shown below as it is)

Accuracy : 0.878





Question 8 (5 Marks)

In all the models above you would have used cross entropy loss. Now compare the cross entropy loss with the squared error loss. I would again like to see some automatically generated plots or your own plots to convince me whether one is better than the other.

Best Configuration used for comparison

Activation : Relu

Optimizer : Adam

Learning Rate : 0.001

Epochs : 20

Batch Size : 128

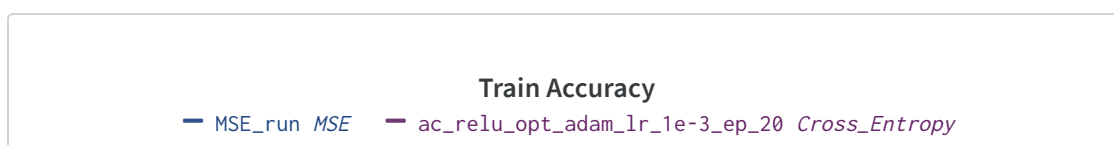
Epsilon : 1e-10

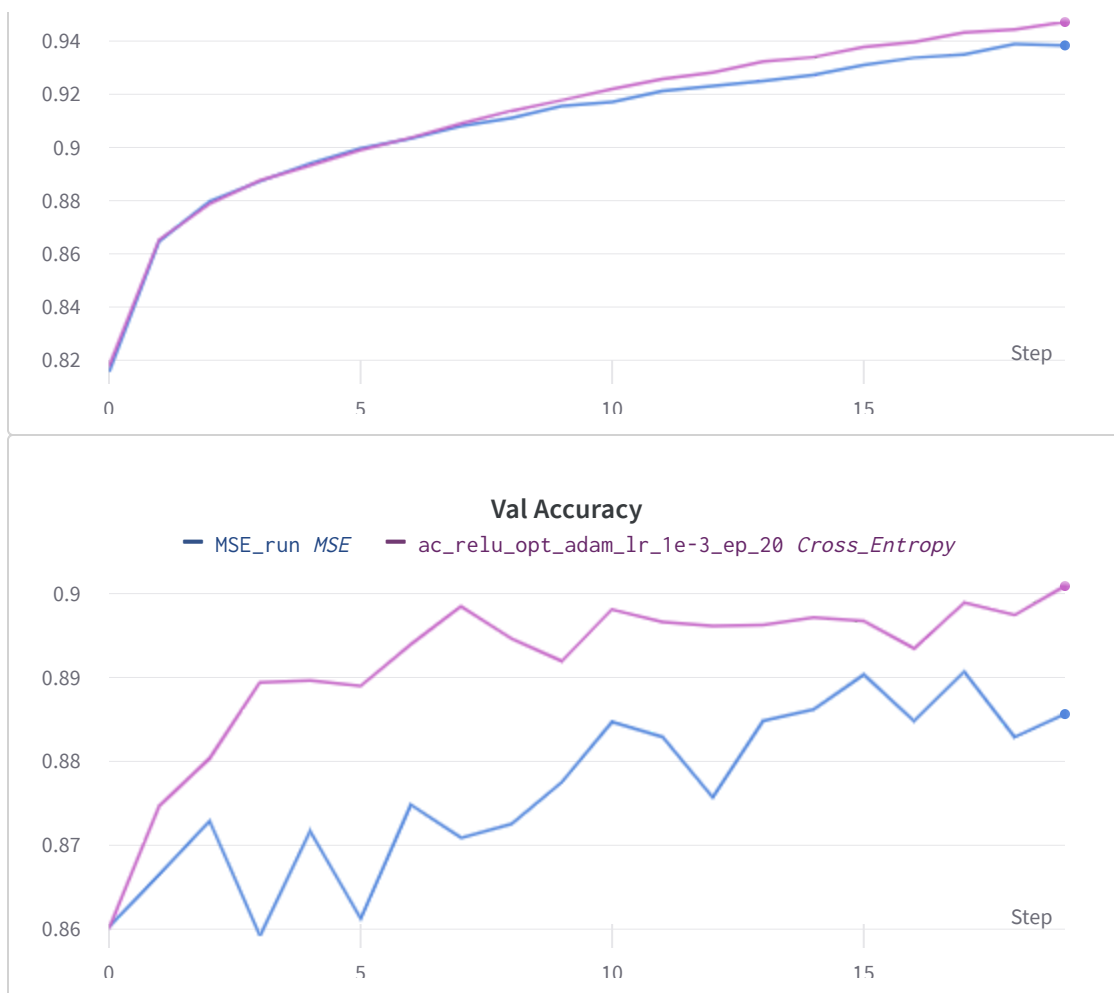
Hidden Layers : 2

Layer Size : 256

Weight Decay : 1e-6

Weight Init : Xavier





▼ Question 9 (10 Marks)

Paste a link to your github code for this assignment

<https://github.com/Chathurvedhi/Deep-Learning-CS6910-A1>

- We will check for coding style, clarity in using functions and a `README` file with clear instructions on training and evaluating the model (the 10 marks will be based on this)
- We will also run a plagiarism check to ensure that the code is not copied (0 marks in the assignment if we find that the code is plagiarized)
- We will also check if the training and test data has been split properly and randomly. You will get 0 marks on the assignment if

we find any cheating (e.g., adding test data to training data) to get higher accuracy

▼ Question 10 (10 Marks)

Based on your learnings above, give me 3 recommendations for what would work for the MNIST dataset (not Fashion-MNIST). Just to be clear, I am asking you to take your learnings based on extensive experimentation with one dataset and see if these learnings help on another dataset. If I give you a budget of running only 3 hyperparameter configurations as opposed to the large number of experiments you have run above then which 3 would you use and why. Report the accuracies that you obtain using these 3 configurations.

Both MNIST and Fashion_MNIST dataset are based on images but Fashion is more complex and harder to get higher accuracies. We can use out best 3 configurations of the final sweep for higher consistency (One of the highest Val_Acc runs is rmsprop but it is one of the only ones in high accuracy)

Configuration 1:

Activation : Relu

Optimizer : Adam

Learning Rate : 0.001

Epochs : 20

Batch Size : 128

Epsilon : 1e-10

Hidden Layers : 2

Layer Size : 256

Weight Decay : 1e-6

Weight Init : Xavier

Accuracy:

Training : 0.9973

Validation : 0.9823

Test : 0.9723

Configuration 2:

Activation : Tanh

Optimizer : Adam

Learning Rate : 0.0001

Epochs : 30

Batch Size : 64

Epsilon : 1e-8

Hidden Layers : 3

Layer Size : 256

Weight Decay : 0

Weight Init : Xavier

Accuracy:

Training : 0.9896

Validation : 0.9776

Test : 0.9702

Configuration 3:

Activation : Relu

Optimizer : Nadam

Learning Rate : 0.0001

Epochs : 40

Batch Size : 32

Epsilon : 1e-9

Hidden Layers : 3

Layer Size : 128

Weight Decay : 1e-4

Weight Init : Xavier

Accuracy:

Training : 0.9992

Validation : 0.9808

Test : 0.9778

Code Specifications

Please ensure you add all the code used to run your experiments in the GitHub repository.

You must also provide a python script called `train.py` in the root directory of your GitHub repository that accepts the following command line arguments with the specified values -

We will check your code for implementation and ease of use. We will also verify your code works by running the following command and checking wandb logs generated -

```
python train.py --wandb_entity myname --wandb_project myprojectname
```

Arguments to be supported

Name	Default Value	Description
<code>-wp, --wandb_project</code>	CS20B021_A1	Project name used to track experiments in Weights & Biases dashboard
<code>-we, --wandb_entity</code>	chathur	Wandb Entity used to track experiments in the Weights & Biases dashboard.
<code>-d, --dataset</code>	fashion_mnist	choices: ["mnist", "fashion_mnist"]
<code>-e, --epochs</code>	20	Number of epochs to train neural network.
<code>-b, --batch_size</code>	128	Batch size used to train neural network.
<code>-l, --loss</code>	cross_entropy	choices: ["mean_squared_error", "cross_entropy"]
<code>-o, --optimizer</code>	adam	choices: ["sgd", "momentum", "nesterov", "rmsprop", "adam", "nadam"]
<code>-lr, --learning_rate</code>	0.001	Learning rate used to optimize model parameters
<code>-m, --momentum</code>	0.9	Momentum used by momentum and nag optimizers.

Name	Default Value	Description
<code>-beta</code> , <code>--beta</code>	0.9	Beta used by rmsprop optimizer
<code>-beta1</code> , <code>--beta1</code>	0.9	Beta1 used by adam and nadam optimizers.
<code>-beta2</code> , <code>--beta2</code>	0.999	Beta2 used by adam and nadam optimizers.
<code>-eps</code> , <code>--epsilon</code>	1e-10	Epsilon used by optimizers.
<code>-w_d</code> , <code>--weight_decay</code>	1e-6	Weight decay used by optimizers.
<code>-w_i</code> , <code>--weight_init</code>	xavier	choices: ["random", "xavier"]
<code>-nhl</code> , <code>--num_layers</code>	2	Number of hidden layers used in feedforward neural network.
<code>-sz</code> , <code>--hidden_size</code>	256	Number of hidden neurons in a feedforward layer.
<code>-a</code> , <code>--activation</code>	sigmoid	choices: ["identity", "sigmoid", "tanh", "relu"]

Please set the default hyperparameters to the values that give you your best validation accuracy. (Hint: Refer to the Wandb sweeps conducted.)

You may also add additional arguments with appropriate default values.

▼ Self Declaration

I, Chathurvedhi Talapaneni_CS20B021, swear on my honour that I have written the code and the report by myself and have not copied it from the internet or other students.

https://wandb.ai/chathur/CS20B021_A1/reports/CS6910-_CS20B021-Assignment_1--VmlldzozODI4MDk5